REMARKS

In the Final Office Action, the Examiner rejected claims 1-24. This response neither amends nor cancels any claims. As such, claims 1-24 remain pending. Applicants respectfully request reconsideration and allowance of the pending claims in view of the following remarks.

Objections to the Claims

In the previously filed Response to the Office Action mailed on September 12, 2007, Applicants amended each of independent claims 1, 8, 15, and 22 to include the distinguishing feature that a configurator loads configuration information from a "*cached* configuration file that *originated from a backend data store*." (Emphasis added). As a general overview, the present application discloses that a web presentation architecture (WPA) controller, in accordance with embodiments of the present invention, may include a configurator that functions to *load required configuration information* into memory at startup time, which may then be used by subsequent HTTP requests received by the server at a later point in time. *See* Application, paragraph [0034]. In response to the client request (e.g., HTTP request), the WPA controller may further include a service manager 134 configured to communicate with one or more backend services 136 in order to obtain the appropriate data corresponding to each client request 148. *See id.* at paragraph [0033]; Fig. 2. Thus, the configuration information required by the configurator may be just one of various types of data which may be obtained through the one or more backend services 136 in response to a client request.

In objecting to the present claims in the Final Office Action, the Examiner alleged that the present application fails to include support for the caching of configuration files, as well as the storing of cached configuration information. Specifically, the Examiner stated:

> ... the data loaded by the configurator is cached and stored as a singleton object, but not the configuration file(s) 212 is cached – please see more in FIG. 2, emphasis added).
>
> The examiner notes that if the Applicants appear to equate "Singleton object" 214 as "cached configuration file", then the subsequent limitations "wherein the configurator stores the configuration information [from said cached configuration file] for subsequent access" would not have any support from the originally filed disclosure (stores said configuration information where?).
>
> That is to say, the originally filed disclosure does not fully support the limitation "a <u>cached</u> configuration file" – emphasis added.
>
> Final Office Action, pages 2-3. (Emphasis in original).

Applicants respectfully traverse the Examiner's objection and assert that the previously submitted amendments to independent claims 1, 8, 15, and 22 are fully supported by the specification.

First, with regard to the Examiner's contention that the present application fails to disclose the *caching* of configuration files, Applicants direct the Examiner's attention to paragraphs [0038]-[0041] of the present application, which generally disclose that a configurator 210, in accordance with one embodiment, may use a servlet to locate appropriate properties files (e.g., by way of the backend services 136), which may be one or more configuration files 212. *See* Application, paragraph [0038]. Each properties file (or configuration file) may be *loaded into the configurator's* *mapper object*. *See id.* at paragraph [0041]. Thus, the present application makes it clear that a mapper object, which is part of a configurator, may include one or more properties or configuration files.

Further, the mapper object, in certain embodiments, may be created or instantiated as a singleton object, which is defined as an object which exists in memory such that only one of that type of object exists at any given time, and wherein the singleton object is *not* destroyed after use, unlike most conventional objects, but rather is kept in memory until accessed again. *See id.* at paragraph [0037]. As the Examiner will appreciate, objects, such as the above-described "mapper" or "singleton" objects, are generally regarded in the present context as instantiations of a particular data structure which is created or *cached* in memory for use by one or more programs during the course of program execution. Indeed, the Examiner appears to have acknowledged this interpretation by hypothetically equating a "singleton object 214 as [a] 'cached configuration file.'" Final Office Action, page 3. Thus, contrary to the Examiner's view, Applicants submit that the present application clearly provides adequate support for the loading of configuration information from *cached configuration files* (e.g., from mapper or singleton objects in memory) which originated from a backend store.

Next, with regard to the Examiner's assertion that the present application fails to provide support for the storing of configuration information for subsequent access, Applicants note that the present application discloses that the WPA controller may further include an object cache manager 114 configured to create customized *in-memory cache* which may be used by the WPA controller for *storing objects having data originating from backend stores*. *See* Application, paragraph [0029]. The object cache manager 114 is described as improving performance by reducing the processing time associated with obtaining data from backend stores or services. *See id.* For instance, rather than requiring the configurator to retrieve the needed configuration files from a backend store *each time* a similar client request is received, the object cache manager 114 may create an in-memory cache for *storing objects for subsequent use in the processing of later requests*. *See id.* Therefore, for subsequent similar client requests, the configuration information may be obtained directly from a mapper object stored an in-memory cache created by the object cache manager 114, thus improving processing time by removing the need to access the backend store for each similar subsequent request. Accordingly, Applicants respectfully submit that the present application clearly provides support for the storing of configuration information.

In view of the foregoing discussion, Applicants assert that the claim features objected to by the Examiner in the Final Office Action are each fully supported by the present application. As such, Applicants respectfully request withdrawal of the objections to independent claims 1, 8, 15, and 22.

<u>Claim Rejections under 35 U.S.C. § 101</u>

In the Final Office Action, the Examiner rejected claims 1-7 and 15-21 under 35 U.S.C. § 101, alleging that these claims are directed to non-statutory subject matter. In particular, the Examiner stated that:

As set forth in the previous Office action mailed September 12, 2007, January 26, 2007 and August 7, 2006, claims 1-7 and 15-21 are rejected because the claimed invention is directed to non-statutory subject matter. They amount to Functional Descriptive Material: "Data Structures" representing descriptive material per se or "Computer Programs" representing computer listings per se.

Claims 1 and 15

Claims 1 and 15 recite "A system for creating web applications ...", which may comprise only software components (i.e., *"a controller generator"* 102 and *"a configurator generator"* 116, which can be implemented as a Servlet, [0019], lines 7-10).

Data structures not claimed as embodied in computer-readable media are descriptive material per se and are not statutory because they are not capable of causing functional change in the computer. See, e.g., Warmerdam, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure per se held nonstatutory). Such claimed data structures do not define any structural and functional interrelationships between the data structure <u>and</u> other claimed aspects of the invention <u>which permit</u> the data structure's <u>functionality to be realized</u>. In contrast, a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure <u>and</u> the computer software and hardware components which permit the data structure's <u>functionality to be realized</u>, and is thus statutory (emphasis added).

Similarly, computer programs claimed as computer listings per se, i.e., the descriptions or expressions of the programs, are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program <u>and</u> other claimed elements of a computer <u>which permit</u> the computer program's <u>functionality to be realized</u>. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory. See Lowry, 32 F.3d at 1583-84, 32 USPQ2d at 1035. Accordingly, it is important to

> distinguish claims that define descriptive material per se from claims that define statutory inventions (emphasis added). See MPEP 2106.01 (I).
>
> **Claims 2-7 and 16-21:**
>
> Claims 2-7 and 15-21 further recite functional descriptions of said software components and do not remedy the deficiencies of independent claims 1 and 15, respectively.
>
> Final Office Action, pages 3-5. (Emphasis in original).

Applicants respectfully traverse the Examiner's rejection of claims 1-7 and 15-21 and assert that the present claims are clearly directed to statutory subject matter. Any analysis of whether a claim is directed to statutory subject matter begins first with the language of 35 U.S.C. § 101, which reads:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title.
>
> 35 U.S.C. § 101.

In interpreting Section 101, the Supreme Court stated that Congress intended statutory subject matter to "include *anything* under the sun that is made by man." *Diamond v. Chakrabarty*, 447 U.S. 303, 309, 206 U.S.P.Q. 193, 197 (1980) (emphasis added). Although this statement may appear limitless, the Supreme Court has identified three categories of unpatentable subject matter: laws of nature, natural phenomena, and abstract ideas. *See Diamond v. Diehr*, 450 U.S. 175, 182, 209 U.S.P.Q. 1, 7 (1981). Accordingly, so long as a claim is not directed to one of the three specific areas listed above, the claim is directed to patentable subject matter. Thus, it is improper to read restrictions into Section 101 regarding subject matter that may be patented where the legislative history does not indicate that Congress clearly intended such a limitation. *In re Alappat*, 31 U.S.P.Q.2d 1545, 1556 (Fed. Cir. 1994) (citing *Chakrabarty* 447 U.S. at 308).

Moreover, the Federal Circuit has held that the mere fact that a claim includes or is directed to an algorithm is no ground for holding that a claim is directed to non-statutory subject matter. *See In re Iwashashi*, 12 U.S.P.Q.2d 1908, 1911 (Fed. Cir 1989). Rather, the proscription against patenting an algorithm, to the extent it still exists, is narrowly limited to *mathematical algorithms in the abstract*, e.g., describing a mathematical algorithm as a procedure for solving a given type of mathematical problem. *See AT&T Corp. v. Excel Communications, Inc.*, 50 U.S.P.Q.2d 1447, 1450 (Fed. Cir 1999). Indeed, the courts are aware that any step-by-step process, be it electronic, chemical, or mechanical, involves an algorithm. *Id.* at 1450.

Thus, an inquiry into what qualifies as statutory subject matter simply requires "an examination of the contested claims to see if the claimed subject matter as a whole is a disembodied mathematical concept representing nothing more than a 'law of nature' or an 'abstract idea, or if the mathematical concept has been reduced to some practical application rendering it 'useful.'" *Id.* at 1451 (citing and quoting *In re Alappat*, 31 U.S.P.Q.2d at 1557). Furthermore, a Section 101 analysis "demands that the focus in any statutory subject matter analysis be on the *claim as a whole*." *In re Alappat*, 31 U.S.P.Q.2d at 1557 (citing *Diehr*, 450

U.S. at 192) (emphasis in original). Indeed, the dispositive inquiry is whether the claim *as a whole* is directed to statutory subject matter; it is irrelevant that a claim may contain, as part of the whole, subject matter that would not be patentable by itself. *Id.*

Independent Claims 1 and 15

Independent claims 1 and 15 each recite "a *system* for creating web applications." (Emphasis added). Independent claim 1 further recites, *inter alia*, "a controller generator ... adapted to provide a web application with a controller that *receives* a request for data ... and *sending* information to the user" and "a configurator generator ... adapted to provide a configurator that *loads* configuration information..." (Emphasis added). Independent claim 15 similarly recites, *inter alia*, "means for creating a controller ... adapted to *receive* a request for data" and "means for creating a configurator that *loads* configuration information..." (Emphasis added). Applicants respectfully disagree with the Examiner's assertions and interpretation of the law under 35 U.S.C. § 101. In particular, Applicants note that the Examiner's rejection appears to be based on the mistaken assumption that the subject matter recited by independent claims 1 and 15 are purely software (e.g., "Data Structures"). *See* Final Office Action, pages 3-5. This assumption is clearly erroneous.

First, Applicants respectfully submit that the "configurator generator" and "controller generator," as recited by independent claim 1, would clearly be understood by one skilled in the art to encompass more than "only software components," as suggested by the Examiner. *See id.* at page 4. For instance, the "configurator generator" is recited as providing a configurator for *loading configuration information.* Similarly, the "controller generator" is recited as providing a controller for *receiving* and *sending* information. Thus, each of these recited elements clearly carry out specific *functions* which could not be accomplished by software alone. That is, software, by itself, is simply a listing of instructions, encoded or otherwise. However, while such instructions may define one or more specific functions, it is generally well understood that *at least some hardware* is required to execute the instructions defined by software in order to perform the functions defined therein. Indeed, Applicants submit that one skilled in the art would clearly understand that the recited controller generator and configurator generator each includes at least some hardware components for carrying out the functions of *loading, storing,* and *receiving* information. Therefore, contrary to the Examiner's view, Applicants respectfully submit that each of the elements recited by independent claims 1 and 15 must encompass *more* than just software. As such, each of independent claims 1 and 15 are believed to be statutory under Section 101.

Further, Applicants direct the Examiner's attention to the preamble of each of independent claims 1 and 15, which recites "a system," and would be readily appreciated by one skilled in the art as encompassing *at least some hardware elements.* Moreover, referring to independent claim 22, Applicants note that claim 22 recites application instructions (i.e., software) stored on a *machine readable medium* for performing the same functionality generally recited in independent claims 1 and 15. Thus, the doctrine of claim differentiation compels an interpretation that the recited systems of claims 1 and 15 include at least some hardware elements and, thus, cannot simply be interpreted as software *per se*. Indeed, as the Examiner will appreciate, a "system" is generally defined as "a group of devices or artificial objects or an organization forming a network esp. for distributing something or serving a common purpose." *Webster's Ninth New Collegiate Dictionary*, page 1199 (1989). Additionally, the present application *clearly* describes embodiments of the disclosed web presentation architecture (WPA) as being "adapted to execute on a <u>processor-based device</u> such as a <u>computer system</u> or the like." Application, paragraph [0017]. (Emphasis added). Further, the system recited by claim 15 includes "means for" language, which should be addressed in accordance with M.P.E.P. § 2181. Accordingly, Applicants believe that the system recited in claim 15 includes sufficient structure (e.g., processor-based device) to comply with the guidelines set forth under M.P.E.P. § 2106.01.

With the foregoing discussion in mind, Applicants respectfully assert that the M.P.E.P. makes it clear that a claim directed towards a "statutory manufacture or machine," such as a processor-based device or computer system, is statutory *even if* certain elements (e.g., the configurator generator and controller generator) may include software. In particular, Applicants direct the Examiner's attention to Section 2106.01(I) of the M.P.E.P. which states, in relevant part:

> Computer programs are often recited as part of a claim. USPTO personnel should determine whether the computer program is being claimed as part of an otherwise statutory manufacture or machine. In such a case, *the claim remains statutory irrespective of the fact that a computer program is included in the claim.* The same result occurs when a computer program is used in a computerized process where the computer executes the instructions set forth in the computer program. Only when the claimed invention taken as a whole is directed to a mere program listing, i.e., to only its

> description or expression, is it descriptive material *per se* and hence nonstatutory.
>
> ***
>
> Since a computer program is merely a set of instructions capable of being executed by a computer, the computer program itself is not a process and USPTO personnel should treat a claim for a computer program, without the computer-readable medium needed to realize the computer program's functionality, as nonstatutory functional descriptive material. When a computer program is claimed in a process where the computer is executing the computer program's instructions, USPTO personnel should treat the claim as a process claim. *When a computer program is recited in conjunction with a physical structure, such as a computer memory, USPTO personnel should treat the claim as a product claim.*
>
> M.P.E.P. § 2106.01(I). (Emphasis added).

Therefore, even though certain elements recited by independent claims 1 and 15 may include software, Applicants submit that claims 1 and 15 are *clearly* statutory under Section 101 because the software elements are being claimed as *part* of a "system." The claims are *not*, as the Examiner suggests, directed towards pure data structures or program listings.

Further, Applicants note that the Examiner's apparent reliance on *In re Warmerdam* in setting forth the present rejection is misplaced. In *Warmerdam*, the Federal Circuit concluded that the applicant's method claims were directed to nonstatutory subject matter because they related to *method steps* involving the mere manipulation of *abstract ideas*. However, with respect to the applicant's claim directed to "a machine," the court stated that those claims were "clearly patentable subject matter." *Warmerdam*, 31 U.S.P.Q. 2d at 1759. (Emphasis added). With this distinction in mind, Applicants submit that the "system" recited by each of independent claims 1 and 15 is *clearly* directed towards a machine (e.g., processor-based device and/or computer system) and, therefore, believed to be clearly statutory in view of *Warmerdam*.

In view of the arguments presented above, Applicants contend that the rejections of independent claims 1 and 15 under 35 U.S.C. § 101 are improper. Accordingly, Applicants respectfully request that the Examiner withdraw the rejection under 35 U.S.C. § 101 and provide an indication of allowance for claims 1 and 15, as well as those claims depending therefrom.

Claim Rejections Under 35 U.S.C. § 102

In the Final Office Action, the Examiner rejected claims 1-24 under 35 U.S.C. § 102(b) as being anticipated by Hutsch et al., U.S. Publication No. 2001/0034771 A1 (hereinafter referred to as "the Hutsch reference"). Specifically, with regard to independent claims 1, 8, 15, and 22, the Examiner stated in relevant part:

> **Claim 1:**
> Hutsch discloses *a system for creating web applications* (e.g., FIG. 3A, [0115]; FIG. 8, [0234-0245]), *the system comprising:*
>
> *a controller generator that is adapted to provide a web application with a controller that receives a request for data from a user and responds to the request by sending information to the user* (e.g., FIG. 8, Web Server 320 provides web applications to Client Browser 304 after receiving HTTP request, [0237-0245]; [0178-0181])*; and*
>
> *a configurator generator that is adapted to provide a configurator that loads configuration information for use by the controller from a cached configuration file* (e.g., [0239], FIG. 8, Configuration Service 336 having configuration information,

[0156]; configuration information stored in user/application profiles, [0310]-[0318]; user/application profiles as XML files, [0321]-[0323]);

that originated from a backend data store (e.g., FIG. 15, Configuration Back End Databases 337, [0326]; Cache 1560 for data originated from the Configuration Back End Databases 337, [0346-0356]); FIG. 8, block 336)*, and*

wherein the configurator stores the configuration information for subsequent access (e.g., FIG. 8, storing the configuration information from user/application profiles in Configuration Service 336 to Profiling Service Configuration File 802 for subsequent access, [0239]).

***

Claim 8:

Hutsch discloses *a method of creating web applications, the method comprising:*

creating, with a processor-based device, a controller that receives a request for data from a user and responds to the request by sending information to the user (e.g., FIG. 8, Web Server 320 provides web applications to Client Browser 304 after receiving HTTP request, [0237-0245]; [0178-0181])*; and*

providing a configurator that loads configuration information for use by the controller from a cached configuration file and wherein the configurator stores the configuration information for subsequent access (e.g., [0239], Configuration Service 336 having configuration information, [0156]; configuration information stored in user/application profiles, [0310]-[0318]; user/application profiles as XML files, [0321]-[0323]);

the cached configuration file that originated from a backend data store (e.g., FIG. 15, Configuration Back End Databases 337, [0326]; Cache 1560 for data originated from the Configuration Back End Databases 337, [0346-0356]); caching configuration information for subsequent access, [0327-03291).

***

Claim 15:

Hutsch discloses *a system for creating web applications, the system comprising:*

means for creating a controller that is adapted to receive a request for data from a user and respond to the request (e.g., FIG. 8, Web Server 320 provides web applications to Client Browser 304 after receiving HTTP request, [0237-0245]; [0178-0181])*; and*

means for creating a configurator that loads configuration information for use by the controller from a cached configuration file and wherein the configurator stores the configuration information for subsequent access (e.g., [0239]; FIG . 8, Configuration Service 336 having configuration information, [0156]; configuration information stored in user/application profiles, [0310]-[0318]; user/application profiles as XML files, [0321]-[0323]; [0239])

the cached configuration file that originated from a backend data store (e.g., FIG. 15, Configuration Back End Databases 337, [0326]; Cache 1560 for data originated from the Configuration Back End Databases 337, [0346-0356]); caching configuration information for subsequent access, [0239], [0327-0329]).

***

Claim 22:

Hutsch discloses *a machine readable medium, comprising:*

a controller generator stored on the machine readable medium, the controller generator being adapted to provide a web application with a controller that receives a request for data from a user and responds to the request by sending information to the user (e.g., FIG. 8, Web Server 320 provides web applications to Client Browser 304 after receiving HTTP request, [0237-0245]; [0178-01 81])*; and*

a configurator generator stored on the machine readable medium, the configurator generator being adapted to provide a configurator that loads configuration information for use by the controller from a cached configuration file and wherein the

> *configurator stores the configuration information for subsequent access* (e.g., [0239], FIG. 8, Configuration Service 336 having configuration information, [OI 561; configuration information stored in user/application profiles, [0310]-[0318]; user/application profiles as XML files, [0321]-[0323]; [0239]);
>
> *the cached configuration file that originated from a backend data store* (e.g., FIG. 15, Configuration Back End Databases 337, [0326]; Cache 1560 for data originated from the Configuration Back End Databases 337, [0346-0356]); caching configuration information for subsequent access, [0239], [0327-0329]).
>
> Final Office Action, pages 5-8, 10. (Emphasis in original).

Applicants respectfully traverse this rejection. Anticipation under Section 102 can be found only if a single reference shows exactly what is claimed. *See Titanium Metals Corp. v. Banner*, 227 U.S.P.Q. 773 (Fed. Cir.1985). For a prior art reference to anticipate under Section 102, every element of the claimed invention must be identically shown in a single reference. *See In re Bond*, 15 U.S.P.Q.2d 1566 (Fed. Cir.1990). That is, the prior art reference must show the *identical invention "in as complete detail as contained in the ... claim*" to support a *prima facie* case of anticipation. *Richardson v. Suzuki Motor Co.*, 9 U.S.P.Q. 2d 1913, 1920 (Fed. Cir. 1989) (emphasis added). Thus, for anticipation, the cited reference must not only disclose all of the recited features but must also disclose the *part-to-part relationships* between these features. *See Lindermann Maschinenfabrik GMBH v. American Hoist & Derrick*, 221 U.S.P.Q. 481, 486 (Fed. Cir.1984). Accordingly, Applicants need only point to a single element or claimed relationship not found in the cited reference to demonstrate that the cited reference fails to anticipate the claimed subject matter.

Features Omitted from Independent Claims 1, 8, 15, and 22

Independent claims 1, 8, 15, and 22, each recite, *inter alia*, "a *configurator that loads configuration information* for use by the controller *from a cached configuration file that originated from a backend store*." (Emphasis added). For example, the present application provides a configuration manager 116 that functions to *oversee the loading* of frequently used information into memory during the startup of a particular web application. *See* Application, paragraph [0030]. As discussed above, frequently used information may include one or more properties or configuration files, which may be cached in memory as a mapper or singleton object. Once the required configuration files are loaded into a mapper object, one or more configurators "may be implemented and each may be responsible *for loading configuration information* related to a specific functionality." *Id.* at paragraph [0036].

The Hutsch reference, to the contrary, does not appear to teach or suggest a configuration manager that provides a configurator for *loading configuration information for use by the controller* from a cached configuration file that originated from a backend store. In setting forth the present ejections, the Examiner appears to have cited the configuration service 336 and the web server 320 of the Hutsch reference as being analogous to the recited configuration manager and controller, respectively. *See* Final Office Action, pages 5-6. However, Applicants are unable to locate where in the Hutsch reference it is disclosed that the configuration service 336 carries out the *loading of configuration information for use by the controller* (e.g., web server 320) *from a cached configuration file that originated from a backend store*.

In sharp contrast, the configuration server 336 appears to merely *create* a configuration file, which is then cached internally with respect to the configuration server 336. *See* Hutsch, Fig. 15 (noting, in particular, cache 1560 and DOM Tree 1570). For instance, the Hutsch reference discloses that once initialized, the configuration server 336 creates a DOM tree 1570 which is populated with system policies from a backend database 337. *See id.* at paragraph [0346]. However, even assuming that the "system policies" from the backend database could be properly construed as "configuration files," the loading of the configuration data stored in the cached DOM Tree 1570 for use by the web server 320 appears to be performed by *external components*. It is not, as the Examiner suggests, performed by the configuration service 336 itself.

To the contrary, the Hutsch reference is clear that when a client request is received by the web server 320, the task of *loading data* requested by the client is delegated to an appropriate universal content provider 331 selected by a universal content broker 113. *See id.* at paragraph [0145]; Fig. 3A. By way of example, when a client request includes a request for *configuration information* or data, a component of the universal content broker, such a configuration proxy 1510 associated with a particular content provider 335, accesses the configuration server to *perform the loading of the necessary configuration data* in response to a particular client request received by the web server 320. *See id.* at paragraph [0326]. Specifically, the configuration proxy 1510 loads or "*gets data from configuration server 336* on behalf of its clients, and *caches the data in a DOM tree in proxy 1510*." *Id.* at paragraph [0327]. (Emphasis added). Once the configuration data from the configuration server 336 is *loaded by the configuration proxy*, the data may be accessed by clients, applications, and components of the web server 320. *See id.*

Indeed, the Examiner has failed to demonstrate that the Hutsch reference discloses that the configuration server 336 is configured to initiate or perform the loading of configuration data (e.g., stored in the DOM tree 1570) for use by a controller (e.g., the web server 320). Instead, the web server 320, as discussed above, relies on one or more configuration proxies 1510 associated with specific content providers 331 for the loading of configuration data which, as *clearly* illustrated in Figs. 3 and 15A of the Hutsch reference, are *separate components* from the configuration service 336. *See id.* at Figs. 3, 15A. As such, Applicants submit that no reasonable analysis of the Hutsch reference could yield an interpretation that the configuration server 336, which the Examiner has asserted as being analogous to the recited configuration generator, performs the function of loading configuration data for use by a controller, as recited by each of independent claims 1, 8, 15, and 22.

For at least this reason, the Hutsch reference cannot anticipate independent claims 1, 8, 15, and 22. Accordingly, Applicants respectfully request withdrawal of the rejection under 35 U.S.C. § 102(b) of independent claims 1, 8, 15, and 22, as well as those claims depending therefrom.

Dependent Claims 2, 9, and 16

Claims 2, 9, and 16 depend from independent claims 1, 8, and 15, respectively, and each generally recite that the configuration file may be a "text properties configuration file." As the Examiner can appreciate, a text properties configuration file, as referred to in the present context, may generally refer to a configuration file which governs the extrinsic appearance of data when it is output or viewed by a user. *See* Application, paragraph [0013]. For instance, a text properties configuration file may govern the extrinsic formats of display of data, which may be irrelevant to the intrinsic characteristics of the data (e.g., the actual content or value of the data itself). *See id.* After careful review, Applicants respectfully assert that the Hutsch reference fails to disclose this recited feature.

In setting forth the present rejections, the Examiner relied solely on the following passage of the Hutsch reference:

> [0029] In still yet another embodiment, the presentation and logic system includes a profiling service and a profiling service configuration file coupled to the profiling service. The profiling service configuration file includes a decision tree wherein the decision tree performs actions. In one embodiment, decision tree is a XML decision tree. The actions performed by the decision tree include: an action based upon request parameters; an action based upon request header parameters; an action based upon user device properties; and an action based upon resource properties.
>
> Hutsch, paragraph [0029].

The above passage appears to describe a profiling service configuration file which may include decision logic for processing a plurality of actions based upon request parameters, device properties, or resource properties. However, Applicants are unable to identify *any* teaching in the cited paragraph which appears to even remotely relate to a configuration file which includes information relating to the properties of textual data. Further, other than merely citing the above passage, the Examiner has provided absolutely no explanation in the Final Office Action as to how the Examiner believes the Hutsch reference discloses a "text properties configuration file." *See generally,* Final Office Action, pages 6-7, 9.

As such, Applicants respectfully submit that the Hutsch reference fails to teach or suggest a configuration file associated with text properties, as recited by claims 2, 9, and 16. Accordingly, Applicants submit that dependent claims 2, 9, and 16 are allowable not only by virtue of their dependencies from independent claims 1, 8, and 15, respectively, but also for the subject matter additionally recited therein.

Dependent Claims 3, 10, and 17

Claims 3, 10, and 17 depend from independent claims 1, 8, and 15, respectively, and each generally recite that configuration information may be stored by the configurator as a singleton object. As discussed above, a singleton object is described in the present application as being:

> ...an object that exists in memory such that only one of that type of object exists at any time in memory. Once created, a singleton object is not destroyed after use, like most objects, but is kept in memory until accessed again.
>
> Application, paragraph [0037].

For instance, once the appropriate properties or configuration file or files are located by a servlet corresponding to a particular configurator class, the configuration information may be stored as a singleton object, also referred to as a "mapper." *See id.* at paragraph [0038]. As the Examiner will appreciate, singleton objects are generally static, so that the configuration information remains consistent for subsequent similar requests. Indeed, the present application notes that configuration information, such as error code tables or logging information, are "*static and do not change* after the web application has started." *Id.* at paragraph [0034]. As the Examiner can appreciate, if configuration information obtained in response to a particular client request and stored in a singleton object were to be modified, then the configuration information may be inaccurate or invalid when accessed again by the configurator when processing a subsequent similar client request. Thus, Applicants respectfully submit that one skilled in the art will readily acknowledge that a singleton object is *static* once created in memory.

In the rejecting claims 3, 10, and 17, the Examiner appears to have cited the DOM tree 1570 of the configuration service 336 as being analogous to the recited "singleton object." *See* Final Office Action, pages 6, 8-9. The Examiner's position appears to rely on one particular embodiment of the invention disclosed in the Hutsch reference, which is described as utilizing a single DOM tree 1570 in the cache 1560 of the configuration server 336. *See* Hutsch, paragraph [0328]. Applicants respectfully disagree with the Examiner's proposed correlation for the following reasons.

First, in sharp contrast to the above-recited "singleton object," the use of a single DOM tree 1570, as disclosed by the Hutsch reference, is for the purpose of *tracking configuration data modifications* by various proxies. *See id.* Specifically, the configuration proxies 1510 are described as functioning to cache the configuration data in the DOM tree 1570 into separate DOM tree structures respective to each configuration proxy 1510. *See id.* at paragraph [0327]. Thereafter, *modifications* to the configuration data, depending on each different client request, are initially carried out on the DOM tree stored in a corresponding configuration proxy 1510. *See id.* The proxy 1510 then reports the configuration *changes* to the configuration server 336, which in turn *implements the modifications to the DOM tree 1570.* Indeed, the Hutsch reference makes it clear that the single DOM tree 1570 in the embodiment disclosed in paragraphs [0327]-[0328] is anything but static.

Second, it appears that each configuration proxy 1510 caches its' own respective "copy" of the original DOM tree 1570 created by the configuration server 336. As discussed above, the separate DOM trees *cached by each of the configuration proxies* 1510 serve to receive configuration data *modifications*, which are later merged with the DOM tree 1570 in the configuration server 336 ("master" DOM tree). *See id.* These teachings would appear to suggest that each *separate* DOM tree must initially be a *copy* of the master DOM tree 1570. That is, each separately cached DOM tree created by each configuration proxy 1510 must initially contain the same configuration information as the master DOM tree 1570 cached by the configuration server 336. Therefore, even though the configuration server 336 may include only a single master copy of the DOM tree 1570, the Hutsch reference clearly discloses that additional copies of the master

DOM tree, which are created and cached by each configuration proxy 1510, also exist *concurrently* in memory. As such, Applicants submit that the Examiner's assertion that the master DOM tree 1570 constitutes a "singleton object" clearly contradicts the plain teachings of the cited reference.

For at least the reasons set forth above, Applicants respectfully assert that the Examiner's assertion that the DOM tree 1570 is analogous to the recited singleton object cannot be supported by the teachings of the Hutsch reference. As such, Applicants respectfully submit that the Hutsch reference fails to teach or suggest storing configuration information as a singleton object, as recited by claims 3, 10, and 17. Accordingly, Applicants submit that dependent claims 3, 10, and 17 are allowable not only by virtue of their dependencies from independent claims 1, 8, and 15, respectively, but also for the subject matter additionally recited therein.

Conclusion

In view of the remarks and amendments set forth above, Applicants respectfully request allowance of the pending claims. If the Examiner believes that a telephonic interview will help speed this application toward issuance, the Examiner is invited to contact the undersigned at the telephone number listed below.

Respectfully submitted,

Date: May 28, 2008

[illegible]

Michael G. Fletcher
Reg. No. 32,777
FLETCHER YODER
7915 FM 1960 West, Suite 330
Houston, TX 77070
(281) 970-4545

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
Legal Department, M/S 35
P.O. Box 272400
Fort Collins, Colorado 80527-2400